Attorney Docket No. 499.058PRV

HIGH-AVAILABILITY APPLICATION PROGRAMMING INTERFACE AND METHOD

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to a U.S. Provisional Patent Application entitled **"HIGH AVAILABILITY COMPUTING SYSTEM AND METHOD,"** attorney docket reference 499.057PRV, filed on even date herewith.

FIELD OF THE INVENTION

This invention relates generally to computer communications, and more particularly to control and communication of highly available computing systems.

COPYRIGHT NOTICE/PERMISSION

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever. The following notice applies to the software and data as described below and in the drawings hereto: Copyright © 2000, Silicon Graphics Incorporated, All Rights Reserved.

BACKGROUND OF THE INVENTION

Parallel computer systems provide economic, scalable, and high-availability approaches to computing solutions. From the point of view of managing computer systems including parallel-processor systems, there is a need for system administration and control in order to obtain desired system operation.

60/189,863

Conventional interfaces for various system-administration applications are unsatisfactory because they can be inconsistent, requiring the user to learn each one of a variety of different system-administration applications. Such system administration applications sometimes are not reliable or secure. Customers want to administer their system from a console of their choice, whether it's running Linux, IRIX, MacOS, Solaris, IRIX, Windows NT, or other OS. Typical system administration applications are hard to use and confusing for the non-technical user. Prerequisites aren't given up front or explained clearly, feedback is not given after each operation, and so forth. Applications that are not localized are difficult for non-English-speaking users to use. Applications that are not localized at the outset can be difficult for developers to retrofit with localization function calls.

Conventionally, scalable parallel computing systems have management control software that is not scalable, and in which there are decreasing economies of scale as the number of computing units in the system increases. As the number of computing units in the parallel computing system increases, the overhead communications traffic increases not just geometrically, but in some cases, exponentially, creating increasing overhead burden in which an increasing amount of processing power is used merely by communicating between the computing units, and a proportionately lesser amount of processing power is available to support the application. Thus, the system management for scalable parallel processing systems have diminishing returns in terms of performance and cost. As a result, there is a need for system management software for scalable parallel processing systems that yields increasing scales of economies as the number of computing units is the system increases.

For further background, the following pointers to on-line sources may be used:

Linuxconf, a system administration API and application:

http://www.solucorp.qc.ca/linuxconf/

Veritas Volume Manager Storage Administrator, a Java-based GUI:

http://mailhub.acsu.buffalo.edu/vm/3.0/ref.pdf

http://mailhub.acsu.buffalo.edu/vm/3.0/gsg.pdf

http://mailhub.acsu.buffalo.edu/vm/3.0/cli.pdf

which are reproduced as attached Appendices L, M, and N.

SGI sysadmdesktop, a Motif-based GUI for personal system administration:

http://fizz.engr.sgi.com/imd/oldrel/sysadm/index.html

Sudo, a program that allows system administrators to give certain users the ability to run some commands as root or as another user while logging the commands and arguments:

http://www.courtesan.com/sudo/sudo.html

Microsoft wizards, step-by-step GUI guides, reminiscent of Rhino tasksets:

http://www.microsoft.com

SUMMARY OF THE INVENTION

The present invention provides solutions to the above-described shortcomings in conventional approaches, as well as other advantages apparent from the description and appendices below.

The present invention provides an API having a client-side API that is consistant across many computing platforms, and a server-side API that can be customized for a particular hardware platform. In some embodiments, the client-side API and code is in Java to provide portability, and the server-side API and code is in C++ customized to a particular platform, such as an SGI high-performance system, in order to provide speed and security.

One aspect of the present invention provides an applications-programming interface (API) for high-availability computing in a computer system having at least one server and at least one client, the API including: a server-side API; a client-side API; a model having server-side priveleged-run (runpriv) and privileged-commands (privcmd) function such that only one program need be setuid-root on the server; an association class and three derived subclasses of ComputedAssoc, ChildAttrAssoc, and ParentAttrAssoc for use in server-side code and client-side code; and a graphical user interface client-side model, wherein the association class monitors a parent item, and the three subclasses are responsible for determining child items to be added, changed and deleted.

Another aspect of the present invention provides an information handling system having stored thereon one of the APIs described above.

Another aspect of the present invention provides a computer-readable storage medium having stored thereon one of the APIs described above.

Another aspect of the present invention provides a single-image method for high-availability computing in a computer system having at least one server and at least one client, the method comprising: providing a server-side API; providing a client-side API; using a model having server-side priveleged-run (runpriv) and privileged-commands (privcmd) function such that only one program need be setuid-root on the server; providing an association class and three derived subclasses of ComputedAssoc, ChildAttrAssoc, and ParentAttrAssoc for use in server-side code and client-side code; and providing a graphical user interface client-side model, wherein the association class monitors a parent item, and the three subclasses are responsible for determining child items to be added, changed and deleted.

Another aspect of the present invention provides an article comprising a computer readable medium having instructions thereon, wherein the instructions, when executed in a computer, create a system for executing the method described above.

The present invention provides the following advantages:

Applications based on the present invention look and behave the same, because they use the same components. Thus users of one application based on the present invention can easily transfer to using another. The present invention is built to be reliable and secure; its runpriv and privileged-commands (privcmd) model means that only one program need be setuid-root on the server.

1. By using Java on the client side, Rhino-based applications can be run on any system having a Java virtual machine (Linux, IRIX, Windows NT, MacOS, and so on). Rhino-based applications can thus run as an applet in a web browser, or as a standalone application.
2. Rhino embodies several principles of interaction that have been identified to confuse users. For details, please see hard-copy document titled, "Rhino Users Model," particularly "The Rhino users model in action" section.
3. Rhino is localized, with all the user-visible strings kept separate from the GUI components, making it easy for Rhino-based applications to be localized.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific exemplary embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical and other changes may be made without departing from the scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

The present invention provides an infrastructure for system administration applications. One embodiment, called “Rhino” and available from Silicon Graphics, Inc. (SGI), the assignee of the present invention, is an infrastructure for building applications that configure, manage, and monitor hardware and software. Rhino provides a common, consistent, task-based, internationalized, secure graphical user interface (GUI). Rhino-based applications have a command-line interface (CLI) that system administrators can use to write scripts. Rhino applications consist of a client-side GUI written in Java, a server-side daemon, server-side command-line interfaces written in C++ or C, and server-side libraries for monitoring the state of the system.

Communications between the client and server are non-blocking and transparent to the application. Encryption is supported (pending export compliance approval) in some embodiments, as are security plugins.

The following SGI products use Rhino:

- FailSafe 2.0 GUI. Subsequent overlays have been shipped as patches, on a roughly quarterly basis since original shipment.
- CXFS GUI. First shipped with IRIX 6.5.6F. Bug fixes were made in IRIX 6.5.7F. Subsequent releases are planned with IRIX 6.5.8F and later releases.
- XVM GUI. In development, with planned initial release with IRIX 6.5.10F.
- Installation replicator for Linux. In development, with planned initial release with SGI Linux 1.3 (SGI ProPack 1.3 for Linux).

Rhino was created to solve the following problems:

- Interfaces for various system administration applications can be inconsistent, requiring the user to learn each.
- System administration applications sometimes are not reliable or secure.
- Customers want to administer their system from a console of their choice, whether it's running Linux, IRIX, MacOS, Solaris, IRIX, Windows NT, or other OS.
- Typical system administration applications are hard to use and confusing for the non-technical user. Prerequisites aren't given up front or explained clearly, feedback is not given after each operation, and so forth.

- Applications that are not localized are difficult for non-English-speaking users to use. Applications that are not localized at the outset can be difficult for developers to retrofit with localization function calls.

The present invention solves the problems in the following ways:

- Rhino-based applications look and behave the same, because they use the same components. Thus users of one Rhino-based application can easily transfer to using another.
- Rhino was built to be reliable and secure; its runpriv and privileged-commands (privcmd) model means that only one program need be setuid-root on the server.
- By using Java on the client side, Rhino-based applications can be run on any system having a Java virtual machine (Linux, IRIX, Windows NT, MacOS, and so on). Rhino-based applications can thus run as an applet in a web browser, or as a standalone application.
- Rhino embodies several principles of interaction that have been identified to confuse users. For details, please see hard-copy document titled, "Rhino Users Model," particularly "The Rhino users model in action" section.
- Rhino is localized, with all the user-visible strings kept separate from the GUI components, making it easy for Rhino-based applications to be localized.

There are other Rhino features that address the problems mentioned in the previous section.

Rhino-based graphical user interfaces help to make system administration applications complete and truly usable by customers (ex., FailSafe, CXFS). Today, system administration applications are not considered professional quality unless they have a well-designed GUI, and Rhino makes it easy for developers to create well-designed GUIs for their system administration applications.

Also, having a common infrastructure makes it less time-consuming and expensive for developers to create additional system administration GUIs. This technology has broad applicability.

Attached Appendix J provides a description of how to use one embodiment of the present invention. Attached Appendix K provides code used to implement one embodiment of the present invention.

Conclusion

The present invention provides an API having a client-side API that is consistant across many computing platforms, and a server-side API that can be customized for a particular hardware platform. In some embodiments, the client-side API and code is in Java to provide portability, and the server-side API and code is in C++ customized to a particular platform, such as an SGI high-performance system, in order to provide speed and security.

One aspect of the present invention provides an applications-programming interface (API) for high-availability computing in a computer system having at least one server and at least one client, the API including: a server-side API; a client-side API; a model having server-side priveleged-run (runpriv) and privileged-commands (privcmd) function such that only one program need be setuid-root on the server; an association class and three derived subclasses of ComputedAssoc, ChildAttrAssoc, and ParentAttrAssoc for use in server-side code and client-side code; and a graphical user interface client-side model, wherein the association class monitors a parent item, and the three subclasses are responsible for determining child items to be added, changed and deleted.

In some embodiments of the API, the ComputedAssoc subclass is for deriving classes that can be computed from values of one or more attributes of monitored child and parent items, and that monitors a parent item from a parent catagory and one or more items in a child catagory that are potential children.

In some embodiments of the API, the ChildAttrAssoc subclass is for deriving classes to represent relationships in which a child item stores selectors of one or more parent items as part of attributes of the ChildAttrAssoc subclass.

In some embodiments of the API, the ParentAttrAssoc subclass is for deriving classes to represent relationships in which a parent item stores selectors of one or more child items as part of attributes of the ParentAttrAssoc subclass.

In some embodiments of the API, the association is implemented in JAVA on the client side, enabling applications to be run on any system having a Java virtual machine.

In some embodiments of the API, the API is localized, with all the user-visible strings kept separate from the GUI components, making it easy for Rhino-based applications to be localized.

Some embodiments of the API further include an AssocFactory class that is a factory class for association objects, wherein AssocFactory methods are used to fulfill requests from remote clients.

In some embodiments of the API, information is obtained about one or more associations and items by: obtaining a handle to an association instance; and obtaining information about item instances.

Another aspect of the present invention provides an information handling system having stored thereon one of the APIs described above.

Another aspect of the present invention provides a computer-readable storage medium having stored thereon one of the APIs described above.

Another aspect of the present invention provides a single-image method for high-availability computing in a computer system having at least one server and at least one client, the method comprising: providing a server-side API; providing a client-side API; using a model having server-side priveleged-run (runpriv) and privileged-commands (privcmd) function such that only one program need be setuid-root on the server; providing an association class and three derived subclasses of

ComputedAssoc, ChildAttrAssoc, and ParentAttrAssoc for use in server-side code and client-side code; and providing a graphical user interface client-side model, wherein the association class monitors a parent item, and the three subclasses are responsible for determining child items to be added, changed and deleted.

Another aspect of the present invention provides an article comprising a computer readable medium having instructions thereon, wherein the instructions, when executed in a computer, create a system for executing the method described above.

Another aspect of the present invention provides an information handling system having stored thereon instructions for executing the method.

Another aspect of the present invention provides a computer-readable storage medium having stored thereon instructions for executing the method.

It is to be understood that the above description is intended to be illustrative, and not restrictive. Although numerous characteristics and advantages of various embodiments of the present invention have been set forth in the foregoing description, together with details of the structure and function of various embodiments, many other embodiments and changes to details will be apparent to those of skill in the art upon reviewing the above description. The scope of the invention should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

WHAT IS CLAIMED IS:

1. An applications-programming interface (API) for high-availability computing in a computer system having at least one server and at least one client, the API comprising:

a server-side API;

a client-side API;

a model having server-side priveleged-run (runpriv) and privileged-commands (privcmd) function such that only one program need be setuid-root on the server;

an association class and three derived subclasses of ComputedAssoc, ChildAttrAssoc, and ParentAttrAssoc for use in server-side code and client-side code; and

a graphical user interface client-side model,

wherein the association class monitors a parent item, and the three subclasses are responsible for determining child items to be added, changed and deleted.

2. The API of claim 1, wherein the ComputedAssoc subclass is for deriving classes that can be computed from values of one or more attributes of monitored child and parent items, and that monitors a parent item from a parent catagory and one or more items in a child catagory that are potential children.

3. The API of claim 1, wherein the ChildAttrAssoc subclass is for deriving classes to represent relationships in which a child item stores selectors of one or more parent items as part of attributes of the ChildAttrAssoc subclass.

4. The API of claim 1, wherein the ParentAttrAssoc subclass is for deriving classes to represent relationships in which a parent item stores selectors of one or more child items as part of attributes of the ParentAttrAssoc subclass.

5. The API of claim 1, wherein the association is implemented in JAVA on the client side, enabling applications to be run on any system having a Java virtual machine.

6. The API of claim 1, wherein the API is localized, with all the user-visible strings kept separate from the GUI components, making it easy for Rhino-based applications to be localized.

7. The API of claim 1, further comprising an AssocFactory class that is a factory class for association objects, wherein AssocFactory methods are used to fulfill requests from remote clients.

8. The API of claim 1, wherein information is obtained about one or more associations and items by:

obtaining a handle to an association instance; and

obtaining information about item instances.

9. An information handling system having stored thereon the API of claim 1.

10. A computer-readable storage medium having stored thereon the API of claim 1.

11. A method of providing a single-image model for high-availability computing in a computer system having at least one server and at least one client, the method comprising:

providing a server-side API;

providing a client-side API;

using a model having server-side priveleged-run (runpriv) and privileged-commands (privcmd) function such that only one program need be setuid-root on the server;

providing an association class and three derived subclasses of ComputedAssoc, ChildAttrAssoc, and ParentAttrAssoc for use in server-side code and client-side code; and

providing a graphical user interface client-side model,

wherein the association class monitors a parent item, and the three subclasses are responsible for determining child items to be added, changed and deleted.

12. An article comprising a computer readable medium having instructions thereon, wherein the instructions, when executed in a computer, create a system for executing the method of claim 11.

13. An information handling system having stored thereon instructions for executing the method of claim 11.

14. A computer-readable storage medium having stored thereon instructions for executing the method of claim 11.

ABSTRACT OF THE DISCLOSURE

An applications-programming interface (API) for high-availability computing in a computer system having at least one server and at least one client, the API including: a server-side API; a client-side API; a model having server-side priveleged-run (runpriv) and privileged-commands (privcmd) function such that only one program need be setuid-root on the server; an association class and three derived subclasses of ComputedAssoc, ChildAttrAssoc, and ParentAttrAssoc for use in server-side code and client-side code; and a graphical user interface client-side model, wherein the association class monitors a parent item, and the three subclasses are responsible for determining child items to be added, changed and deleted.

"Express Mail" mailing label no. EL 510 128 107 US
Date of Deposit: 16 March 2000
I hereby certify that this paper or fee is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and in an envelope addressed to Assistant Commissioner for Patents, Washington, D.C. 20231.

Charles A. Lemaire
(Name)

[illegible] 16 March 2000
(Signature) (date)

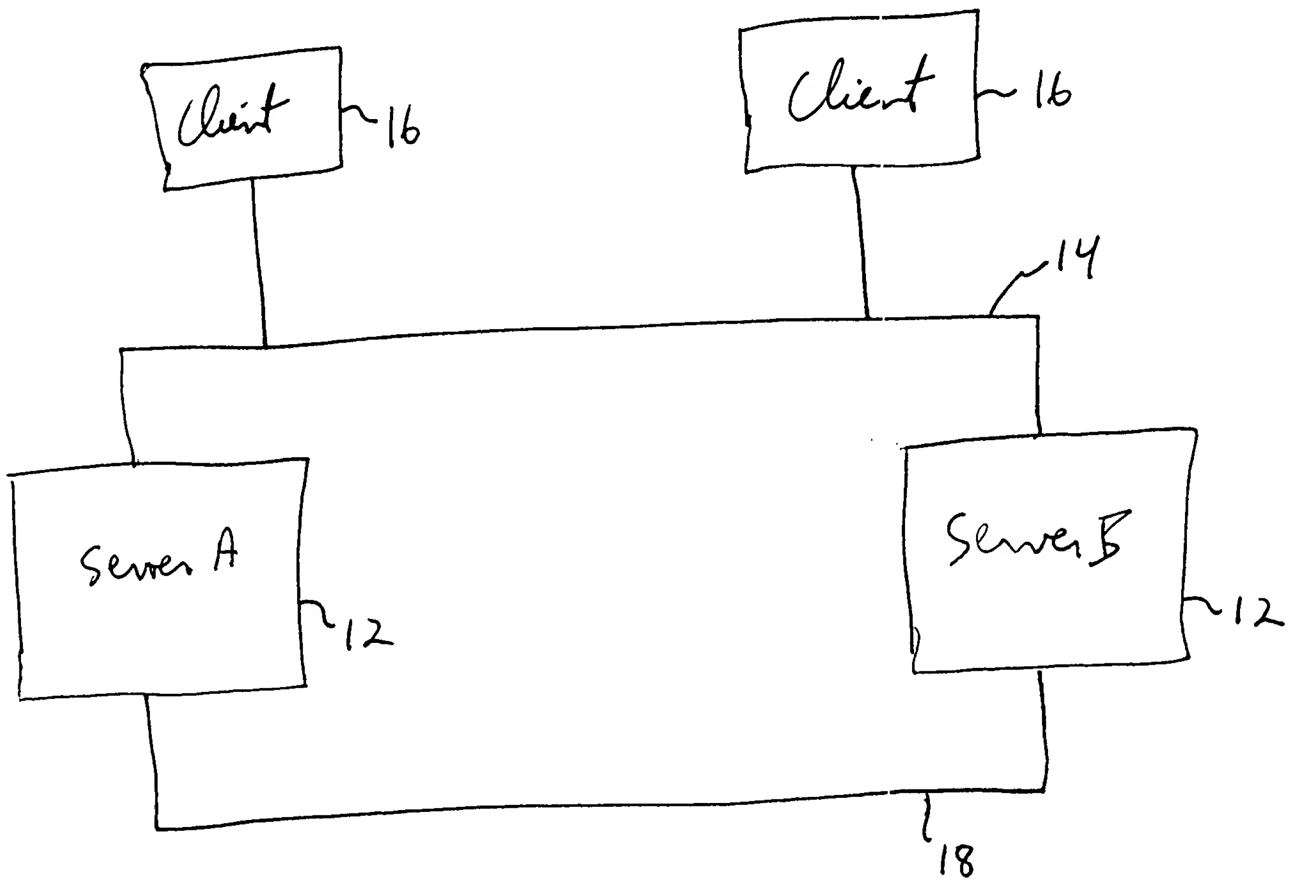
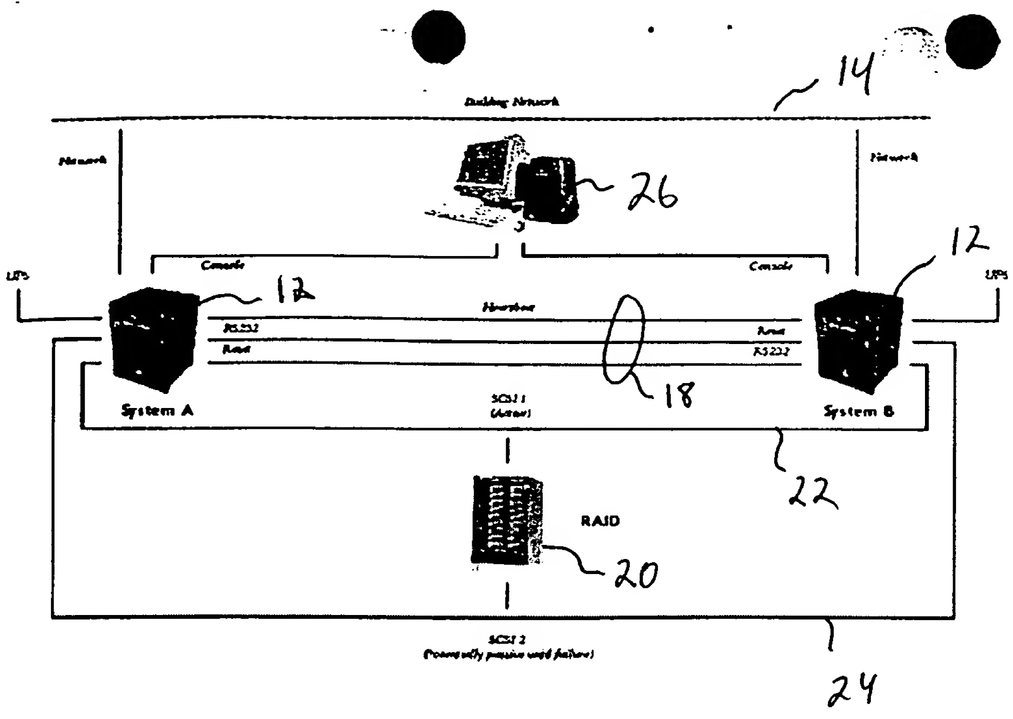
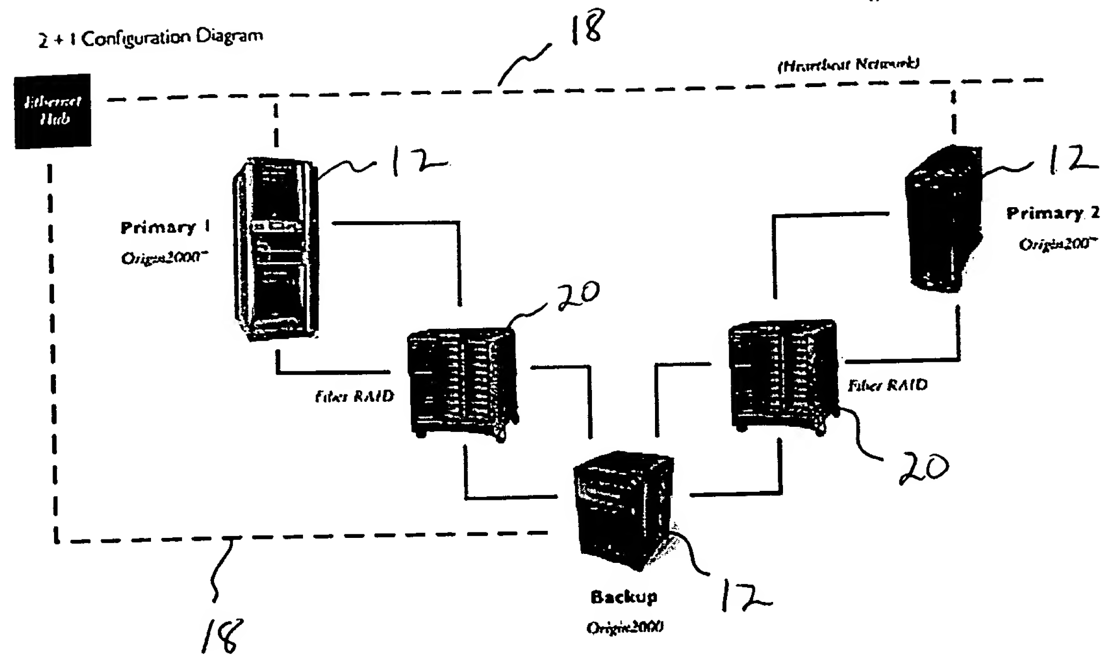


Fig. 1



14
Building Network
Network
Network
26
Console
Console
UPS
UPS
12
12
Heartbeat
RS232
Reset
Reset
RS232
18
System A
System B
SCSI 1 (Active)
22
RAID
20
SCSI 2 (Potentially passive until failure)
24

Fig. 2



2 + 1 Configuration Diagram
18
(Heartbeat Network)
Ethernet Hub
12
Primary 1
Origin2000
12
Primary 2
Origin2000
20
Fiber RAID
Fiber RAID
20
Backup
Origin2000
12
18

Fig. 3

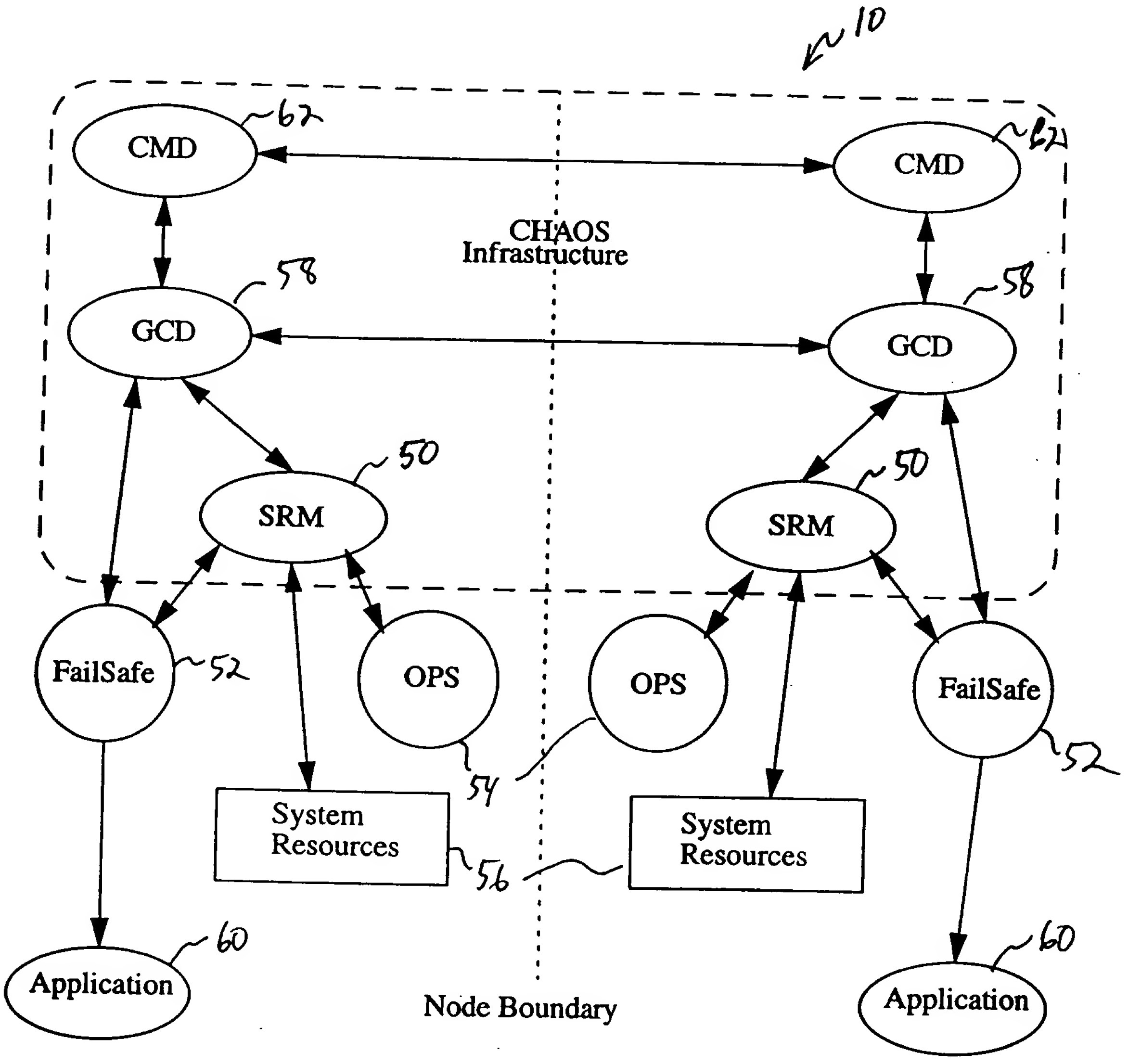

10
CMD
62
CMD
62
CHAOS Infrastructure
GCD
58
GCD
58
SRM
50
SRM
50
FailSafe
52
OPS
OPS
54
FailSafe
52
System Resources
56
System Resources
Application
60
Node Boundary
Application
60

Fig. 4

TABLE OF CONTENTS

Introduction to the Cluster Configuration Database

Rob Bradshaw

Revision 15
Last Update: 3 February 1998

Contents

Introduction

The various pieces of cluster software each have a number of configurable parameters that must be set by the system administrator. In the past these parameters had been maintained in several ASCII text files. However, this is no longer adequate for several reasons:

- A cluster may now consist of 8 or more nodes, which could make a text configuration file large and awkward to work with.
- The configuration information must generally be available to all of the nodes in a cluster. Historically this has been dealt with by manually copying the configuration file from one node to the other, but this will no longer be practical if there are many nodes in a cluster.
- It will soon be necessary to be able to update the configuration from other programs (e.g. Cluster Manager). It is difficult to maintain the validity of data handled in this way if it can

Cluster Administration Interface Model

Rob Bradshaw

Revision 4
Last Update: 22 April 1998

Contents

Introduction

The configuration and administration of SGI cluster systems will be done in at least two different ways:

1. Using a graphical user interface, known tentatively by names such as "Cluster Manager" or "FailSafe Manager". This will be referred to as "The GUI" and will be provided by the System Administration group.

2. Using a command line interface, similar in style to tools such as `xlv_mgr`. This will be referred to as "The CLI" and will be provided by the HA Infrastructure group.

To avoid duplicated effort, both of these tools will use the same mechanisms for effecting changes and monitoring status in the cluster. This document describes those common mechanisms at an architectural level. Specific implementation details are described *[elsewhere]*.

Commands

Most cluster operations ultimately involve the relatively small, discrete tasks which are described in the Cluster Administration Tasks document. Each task will be performed by one of a number of specialized unix *task commands* (provided by the HA Infrastructure group) that will be executed in a separate process by the GUI or CLI.

Cluster Administration Services Basic Services

Rob Bradshaw

Revision 3
Last Update: 28 May 1998

Changes from revision 1 to revision 2 are in this color
Changes from revision 2 to revision 3 are in this color

Contents

Introduction

Cluster Administration Services (CAS) provides a variety of different services, both natively (e.g. CAS messaging) and in the form of plugin services (e.g. the Cluster Adminstration Monitor [CAM]). Underlying these services is a low-level infrastructure that provides such features as logging and storage management. These are referred to collectively as the "CAS Basic Services".

This document describes these basic services provided by CAS and the API used to access them. For information on other aspects of CAS, see:

- Cluster Administration Services - Messaging
- Cluster Administration Services - Plugins

COPY

Cluster Administration Services Messaging

Rob Bradshaw

Revision 6
Last Update: 8 June 1998

Changes from revision 3 to revision 4 are in this color
Changes from revision 4 to revision 5 are in this color
Changes from revision 5 to revision 6 are in this color

Contents

Cluster Administration Services Plugins

Rob Bradshaw

Plugin API Version 1.0.0

Document Revision 3
Last Update: 28 May 1998

Changes from revision 1 to revision 2 are in this color
Changes from revision 2 to revision 3 are in this color

Contents

Introduction

By itself, Cluster Administration Services (CAS) is little more than a message bus, routing data between different pieces of cluster software. Rather than provide additional features on its own, CAS relies on *plugin modules* to extend its functionality. CAS plugin modules (also referred to simply as *plugins*) not only make new services available to client programs, but also provide new communication mechanisms and message types as

Cluster Configuration Database: Back End Interface

Rob Bradshaw

Version 1.0 - Revision 20
Last Update: 2 February 1998

Contents

SiliconGraphic
Computer Systems

Chaos Membership Services

Luca Andrea Castellano
Sharad Srivastava

60199863.031600

Revision No. - 1.1
Date of Last Revision - 03/27/97

SiliconGraphics
Computer Systems

System Resource Manager

Functional Specification, Architecture, & Design

Michael Nishimoto, Paddy Sreenevasan, Manish Verma

Silicon Graphics, Inc.
2011 North Shoreline Boulevard
Mountain View, CA 94043-1389

(Revision 1.04)